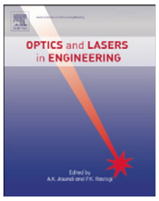
Optics and Lasers in Engineering 50 (2012) 1450–1458

Contents lists available at SciVerse ScienceDirect

Optics and Lasers in Engineering

journal homepage: www.elsevier.com/locate/optlaseng



Accurate and rapid alignment of laser scanned 3D surface using TSK-type neural-fuzzy network-based coarse-to-fine strategy

Jyun-Wei Chang, Sheng-Fuu Lin*, Chi-Yao Hsu

Department of Electrical Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 300, ROC

ARTICLE INFO

Article history:
Received 15 January 2012
Received in revised form
12 April 2012
Accepted 18 April 2012
Available online 24 May 2012

Keywords:
Three-dimensional surface
TSK-type neural-fuzzy network
Principal component analysis
Coarse-to-fine alignment approach

ABSTRACT

Aligning a laser scanned three-dimensional (3D) surface is considered a critical step in object recognition, shape analysis, and automatic visual inspection. Two major concerns for the alignment task are execution time and alignment accuracy. Recently, neural network-based methods have become very popular due to their high efficiency. However, such methods experience difficulty in reaching high accuracy because the use of principal component analysis (PCA) to perform coarse alignment causes a large alignment error. Thus, a TSK-type neural-fuzzy network (TNFN)-based coarse-to-fine 3D surface alignment scheme is proposed in the current paper. Compared with traditional neural network-based approaches, the proposed method provides a coarse-to-fine alignment approach to ensure the accurate pose estimated by TNFN in the coarse phase, as well the high alignment speed provided by TNFN-based surface modeling in the fine phase. Experimental results demonstrate the superior performance of the proposed 3D surface alignment system over existing systems.

1. Introduction

The problem of 3D surface alignment has been implemented by several methods [1–11]. In [1,2] the authors proposed a coarse-to-fine technique. In [3], Ghafoor et al. presented the new simple image-matching algorithm, robust image matching algorithm, an extension to distance transform chamfer matching. In [4,5], authors proposed the alignment systems based on Iterative Closest Point algorithm. In [6], authors present a framework based on a canonical genetic algorithm. Malassiotid and Strintzis propose a local surface descriptor and applied to the problem of aligning partial views of a 3D object [7]. Johan et al. propose a novel Minimum Projection Area-based (MPA) alignment method for pose normalization [8]. Li and Johan propose [9] a 2D sketch-3D model alignment algorithm using view context and shape context matching. In [10,11], authors proposed a novel topological structured pattern, dubbed a spiral facet, and applied for the encoding and alignment of 3D facial triangular mesh surfaces. Among them, a coarse-to-fine technique is a useful way for performing 3D surface alignment [1,2]. Coarse alignment provides an approximate transformation for aligning two surfaces. Such alignment must be efficient and accurate. Fine alignment uses the initial gauss of a transformation given by a coarse alignment as a starting point to iteratively minimize the distance between the input and the destination surfaces. Specifically, in consideration of coarse surface alignment, common neural network-based methods [1,2] utilized PCA [12] for coarsely aligning two surfaces due to its high-speed performance. However, PCA cannot ensure that the laser scanned point clouds have the same orientation of principal axes as the reference model. This phenomenon would cause a high alignment error in the coarse alignment phase. In consideration of traditional fine alignment methods, iterative closet point (ICP) [13] is a typical method to iteratively calculate the rigid-body transformation to minimize the cost function. Although ICP can provide highly accurate 3D surface alignment, its heavy computational cost in searching corresponding points has been criticized by many researchers [1,2,14–16].

For coarse alignment of a 3D surface, to improve the drawback generated by PCA, the present study proposes a TNFN-based coarse alignment approach. Such approach captures VFH of multi-views of a 3D object as the input of a TNFN. The desired output of TNFN is the corresponding pose of the captured feature. Thus, the desired output and the feature input can be utilized for training TNFN. Once TNFN has been trained, inputting VFH of an arbitrary view of an object into the trained TNFN can yield an estimated pose. The estimated output pose can then be utilized to recover the input point clouds to coarsely align with the reference model. The major advantage of the proposed coarse surface alignment method is that TNFN training can infer the relation between the input feature and output pose. This relation results in accurate pose estimation of the input point clouds.

For fine alignment of a 3D surface, a TNFN-based fine alignment approach is proposed in the present work to address

* Corresponding author. Tel.: +886 3 5712121x.54365; fax: +886 3 5715998.
E-mail addresses: jwjhang.ece96g@g2.nctu.edu.tw (J.-W. Chang), sflin@mail.nctu.edu.tw (S.-F. Lin), hsuchyao@yahoo.com.tw (C.-Y. Hsu).

0143-8166/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.optlaseng.2012.04.005

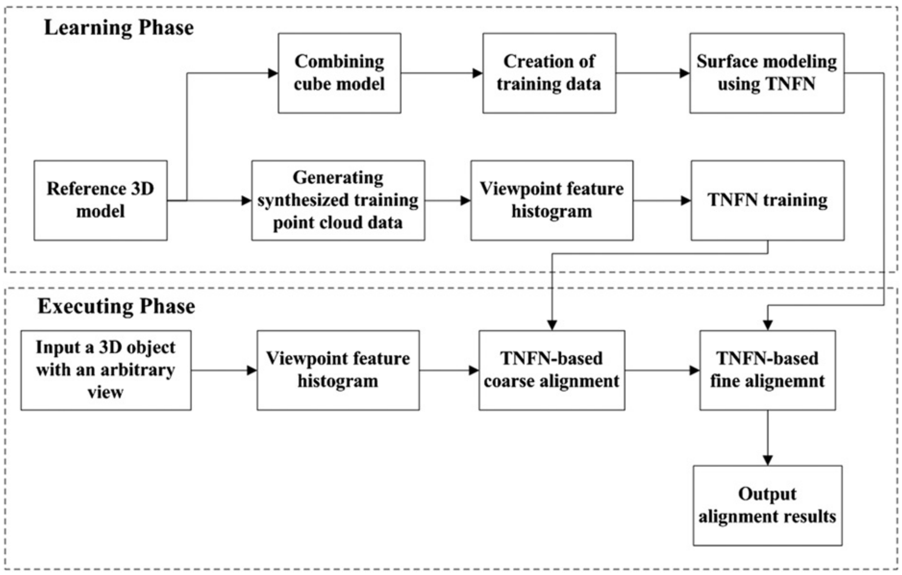

Learning Phase
Reference 3D model
Combining cube model
Creation of training data
Surface modeling using TNFN
Generating synthesized training point cloud data
Viewpoint feature histogram
TNFN training
Executing Phase
Input a 3D object with an arbitrary view
Viewpoint feature histogram
TNFN-based coarse alignment
TNFN-based fine alignemnt
Output alignment results

Fig. 1. Flow diagram of the proposed 3D surface alignment system.

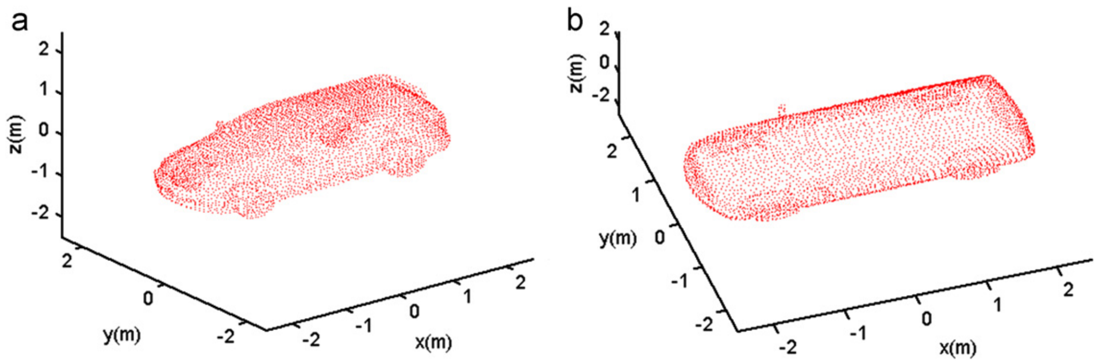


Fig. 2. Point cloud data of the reference model: (a) Front view and (b) Top view.

the drawbacks caused by ICP. The proposed method combines the TNFN-based surface modeling with the downhill simplex optimization method to iteratively reduce the distance from the input 3D surface (i.e., 3D point cloud data [17]) to the reference surface. Due to the TNFN-based surface modeling, the calculation of the corresponding points can be avoided, which increases the alignment speed for the fine alignment task. Evidence of fast execution time in the proposed method can be found in the experimental results.

The rest of this paper is organized as follows. In Section 2, the proposed methodology of 3D surface alignment system is introduced. The experimental results are discussed in Section 3. The conclusion is presented in the last section.

2. Methodology of 3D surface alignment system

Fig. 1 presents the flow diagram of the proposed 3D surface alignment system. In the learning phase, two data flows are performed for training TNFNs to adapt two levels of surface alignment: one for coarse alignment and the other one for fine alignment. In the executing phase, the trained TNFNs are utilized to implement a coarse-to-fine 3D surface alignment task. These two phases are explained in detail to show the process of surface alignment works.

2.1. Learning phase

The objective of the learning procedure is to train two TNFNs for applying coarse-to-fine alignment. The two major parts of the procedure are the coarse alignment learning and the TNFN-based surface modeling. These parts are described in the following subsections.

2.1.1. Coarse alignment learning

The goal of coarse alignment is to determine an approximate rigid transformation that coarsely aligns the reference model with the input point clouds. The coarse alignment must be quick to provide a good initial transformation for the fine alignment task. Thus, TNFN is utilized to learn any case of rigid transformation within the predefined range. Once the learning of TNFN is completed, input arbitrary view of point clouds yield the estimate pose with respect to the reference model. Therefore, the executing phase of the TNFN is simple and efficient.

The procedures of proposed coarse alignment learning involves generating synthesized training point cloud data,

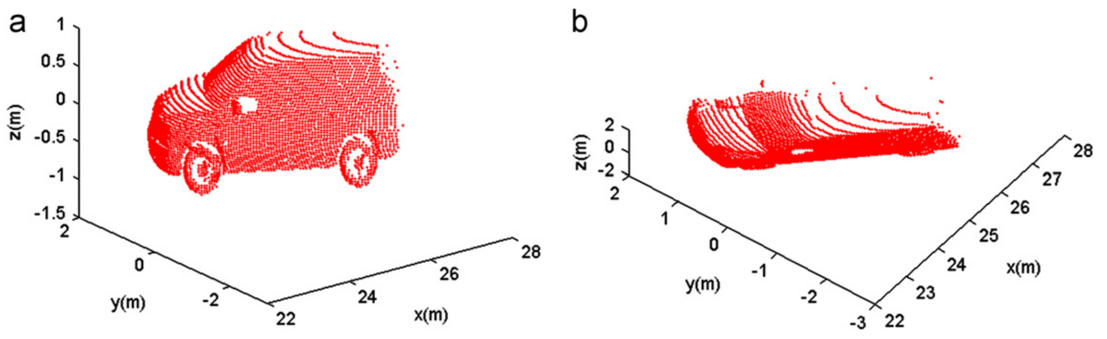


Fig. 3. Example of the simulated training data: (a) Front view and (b) Top view.

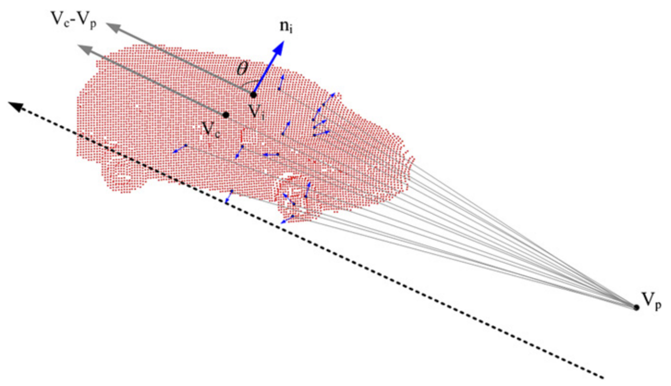


Fig. 4. Creation of viewpoint feature histogram.

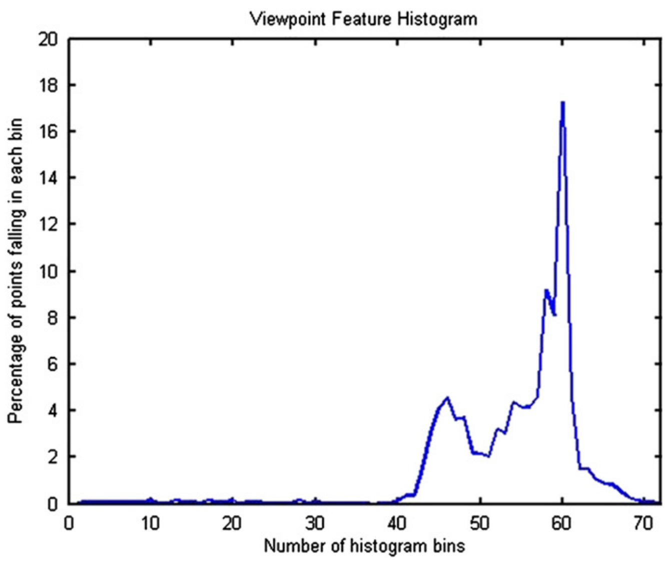


Fig. 5. Example of viewpoint feature histogram for a 3D object appearing in a certain view.

yielding the viewpoint feature histogram, and training the TNFN. These operations are introduced as follows:

(1) Generating synthesized training point cloud data
Fig. 2 depicts the point cloud data of the reference model. The reference model is an integrated model constructed by collecting multi-views of point cloud data. To generate the synthesized training point cloud data, various combinations of translation and rotation transformations within a predefined range are applied in the reference model. The transformation can be considered a rigid transformation, which can be written as follows:

$$m=\begin{bmatrix}x'\\y'\\z'\end{bmatrix}=R\cdot s+T=\begin{bmatrix}r_{11}&r_{12}&r_{13}\\r_{21}&r_{22}&r_{23}\\r_{31}&r_{32}&r_{33}\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\end{bmatrix}+\begin{bmatrix}t_x\\t_y\\t_z\end{bmatrix}, \tag{1}$$

where R is a rotation matrix, T is a translation vector, s is an original set of point cloud data and m is a transformed set of point cloud data. Furthermore, to simulate the real case in a 3D scene, point cloud data that cannot be seen in the viewpoint direction are eliminated. Fig. 3 presents an example of the simulated training data. As shown in this figure, the point cloud data is only a partial of reference model and the unseen point clouds have been eliminated. Therefore, after the training point data has been generated, the following operation is to extract the feature of the point cloud data.

(2) Viewpoint Feature Histogram
VFH was presented by Rusu et al. [18] to show its computationally efficient 3D feature. VFH is computed by accumulating a histogram of the angles between the central viewpoint direction and each normal of point cloud. Fig. 4 illustrates the idea of VFH.

Suppose the central point is V_c and the viewpoint is V_p. Then the central viewpoint direction is V_c-V_p. Thus the angle θ between the central viewpoint direction (V_c-V_p) and each normal n_i of point cloud V_i can be computed as following equation:

$$\theta=\cos^{-1}\left(\frac{(V_c-V_p)\bullet n_i}{||V_c-V_p||\cdot||n_i||}\right). \tag{2}$$

Thereafter, the N-bin orientation histograms (each bin cover 180/N degree) can be calculated by accumulating the angle described in Eq. (2). The histogram in each bin is normalized by dividing the total number of point clouds. Thus, such histogram indicates the percentage of point clouds falling in each bin. Fig. 5 shows an example of VFH for a 3D object appearing in a certain view.

(3) TNFN Training
After extracting VFH from multi-views of the 3D object, let VFH be the input neurons of TNFN and let the corresponding pose be the output neurons of TNFN. The corresponding pose comprises six degrees of freedom, including three rotation angles (ϕ, φ, θ) and three translation parameters (x, y, z).
The structure of a TNFN is shown in Fig. 6, where n represents the dimension of the input. It is a five-layer network structure. In the TNFN, the firing strength of a fuzzy rule is calculated by performing the following ''AND'' operation on the truth values of each variable to its corresponding

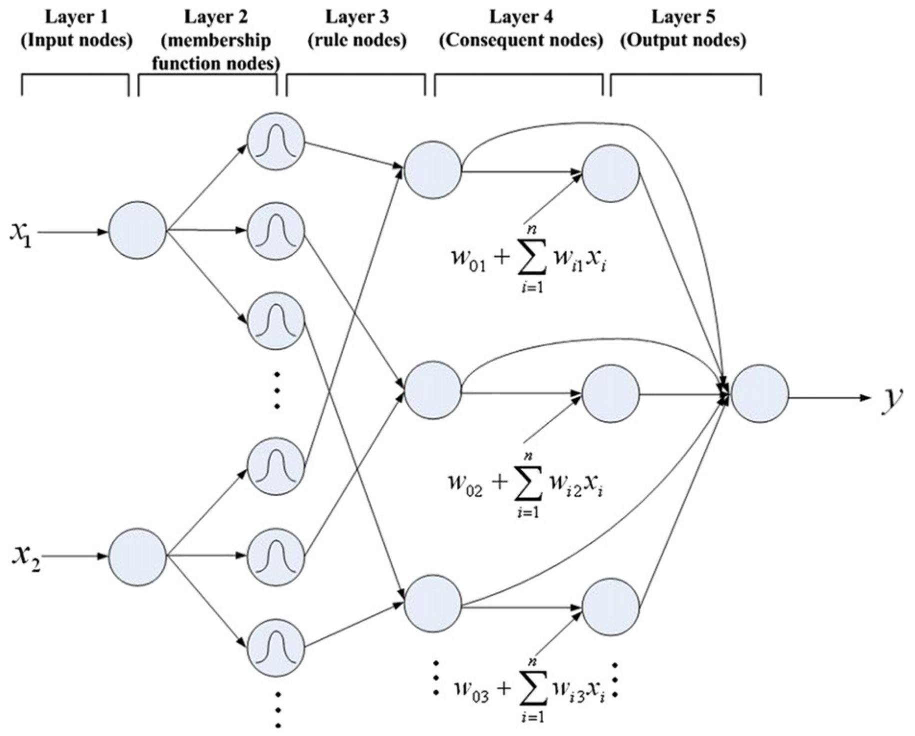

Layer 1 (Input nodes)
Layer 2 (membership function nodes)
Layer 3 (rule nodes)
Layer 4 (Consequent nodes)
Layer 5 (Output nodes)
x_1
x_2
y
$w_{01}+\sum_{i=1}^{n} w_{i1}x_i$
$w_{02}+\sum_{i=1}^{n} w_{i2}x_i$
$w_{03}+\sum_{i=1}^{n} w_{i3}x_i$

Fig. 6. Structure of TNFN.

fuzzy sets:

$$u_{ij}^{(3)} = \prod_{i=1}^{n} \exp\left(-\frac{[u_i^{(1)} - m_{ij}]^2}{\sigma_{ij}^2}\right), \tag{3}$$

where $u_i^{(1)} = x_i$ and $u_{ij}^{(3)}$ are the outputs of 1st and 3rd layers, respectively, and m_{ij} and σ_{ij} are the center and the width of the Gaussian membership function of the jth term of the ith input variable x_i, respectively.

The output of the fuzzy system is computed by:

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{M} u_j^{(3)}}, \tag{4}$$

where $u^{(5)}$ is the output of 5th layer, w_{ij} is the weighting value with ith dimension and jth rule node, and M is the number of a fuzzy rule. Here the dimension of the output is set as six. These outputs are represented as rotation angles (ϕ, φ, θ) and translation parameters (x, y, z).

In this study, data-mining-based evolutionary learning algorithm (DMELA) [19] is adopted for the TNFN training. In DMELA, each individual represents only a partial solution and a full solution is built by means of cooperating with other partial solutions. Thus, each individual can be evolved locally and recombined it with other well-performed individuals to form a good total solution. To concern with fitness evaluation, the fitness of an individual is calculated by summing all combinations of that individual with other individuals and dividing by the total number of combinations. Thus, the fitness value reflects an average value of combined full solutions.

To consider the training of TNFN, there are nine basic steps of DMELA and they are described as follows.

Step1. Initialization: in this step, all fitness values are clear and all genes of individuals are assigned a random value within a predefined range.

Step2. Selection: randomly select n individuals from the population.

Step3. Create a neural network: use the selected n individuals to build a neural network.

Step4. Evaluate the network: after the neural is created, the evaluation is performed according the given problem.

Step5. Selection times check: each individual must be selected sufficient times. If the selection time does not satisfy, then go to Step2 to continue the selection step.

Step6. In this step, the average fitness value of an individual is computed by dividing the total fitness value of each chromosome by the number of times that it has been selected to build networks.

Step7. Termination check: check the fitness value with respect the whole network not a single individual. If the fitness value of the whole network satisfies the pre-setting value, then DMELA terminate.

Step8. Crossover: a two-point crossover strategy is used to exchange the site's values between the selected sites of individual parents to create new individuals, which are offspring inheriting the parents' merits.

Step9. Mutation: in the last step, the gene is mutated at the predetermined rate drawn randomly from the domain of the corresponding variable.

2.1.2. *TNFN-based surface modeling*

The purpose of the TNFN-based surface modeling is to provide an evaluation method for performing the fine alignment of 3D surface. The evaluation is to measure how close distance from the reference surface to input point clouds is. Thus, the major part of the TNFN-based surface modeling is to use the TNFN to model the 3D surface that maps the 3D

Euclidean input space into 1D Euclidean output space. Such mapping can be considered a cost function that evaluates the distance between the input point clouds and the reference model. Thus, the TNFN mapping can combine with the downhill simplex optimization method to iteratively compute the rotation matrix R and translation vector T to perform the fine alignment of 3D surface. The detail of the combination of the TNFN mapping and the downhill simplex optimization will be discussed in the executing phase.

The procedures of modeling the 3D surface involve combining the cube model, creation of training data, and surface modeling using TNFN. These operations are explained bellow.

(1) Combing cube model
To model the reference surface, uniform distributed point clouds are needed to prepare the training data. In the present study, a cube model is generated to be combined with the reference model. The cube model encloses the reference surface, and the point clouds within the cube are sampled uniformly. Thus, the point clouds around the reference model can serve as the training data for modeling the reference surface. Fig. 7 depicts the locations of cube and reference model where the reference model is located at the center of the cube.

(2) Creation of training data
In the creation of training data, we extract the point clouds enclosed the cube satisfying the distance from a point (x, y, z) to the reference model less than a predefined value. The predefined value is set by observing the maximal alignment error yielded from the coarse alignment case. For instance, if the maximal alignment error is 1m, then we can set the predefined value as 1m. Therefore, the point clouds (x, y, z) satisfies

$$\mathrm{Dist}(x,y,z) \leq \text{ predefined value} \tag{5}$$

will be used for training the TNFN.

(3) Surface modeling using TNFN
Similar to the TNFN learning in the coarse alignment learning case, the structure shown in Fig. 6 is used to model the 3D surface. The input of the TNFN is defined as the coordinates (x, y, z) of a point cloud, and the output of the TNFN is the unsigned distance from a point (x, y, z) to the reference model. Thus, the surface of the reference model can be modeled using the TNFN to map the 3D coordinate of point cloud data into the 1D distance between the cube data and the reference model. The representation of the modeling function can be written as follows:

$$\mathrm{Dist} = f(x,y,z). \tag{6}$$

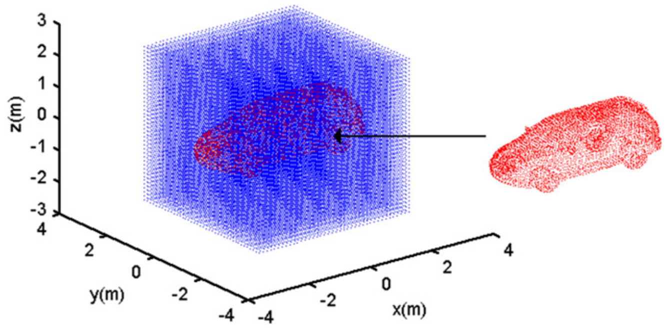


Fig. 7. Location of cube and reference model.

The average distance between the cube data and the reference model can be computed as follows:

$$\mathrm{TotDist} = \frac{1}{N}\sum_{i=1}^{N} f(x_i,y_i,z_i), \tag{7}$$

where N is the number of the cube model. Thus, when the resolution of the cube model is sufficiently high, any arbitrary point clouds within the cube can be send into a trained TNFN to calculate the distance between the input point clouds and the reference model.

In consideration of training a TNFN to model the reference surface, as well as the coarse alignment learning, DMELA [19] is also utilized to perform training the TNFN.

2.2. Execution phase

In the execution phase, the input point clouds are aligned with the reference model by means of VFH extraction, TNFN-based coarse alignment, and TNFN-based fine alignment. VFH extraction has been discussed in Section 2.1.1, whereas the TNFN-based coarse and fine alignments are described as bellow.

2.2.1. TNFN-based coarse alignment

Assuming the VFH descriptor has been calculated, the descriptor is forwarded to the trained TNFN to obtain the rotation angles $(\phi,\ \varphi,\ \theta)$ and translation parameters $(x,\ y,\ z)$. Then, the six parameters are used to compute the rotation matrix R and translation vector T defined in Eq. (1). Based on R and T, we obtain the estimated pose to coarsely align the input point clouds with the reference model.

2.2.2. TNFN-based fine alignment

The procedure of the TNFN-based fine alignment consists of the TNFN mapping and the downhill simplex optimization. In the TNFN mapping, the TNFN maps each 3D point cloud (x, y, z) into a 1D distance function $f(x,\ y,\ z)$ (defined in Eq. (6)). The total distance function $\sum f(x,\ y,\ z)$ is computed by summing of each distance mapping of 3D point cloud. Thus, the total distance function is used as the cost function of the subsequent downhill simplex optimization. In downhill simplex optimization, iterative calculation of rigid transformation between input point clouds and reference model is adopted to minimize the cost function. Each iterative loop uses the downhill simplex method to compute the rotation matrix R and translation vector T to perform fine alignment. Once the downhill simplex optimization is completed, the final R and T are used to calculate the estimate pose that align the input point clouds with reference surface.

Detail steps of the downhill simplex optimization for fine alignment of 3D surface are described as follows:

Step 0: Under 3D rigid body transformation, we choose six degrees of freedom (three rotation angles $(\phi,\ \varphi,\ \theta)$ and three translation parameters (x, y, z) as the vertex of simplex. Then we randomly generate 6+1 initial vertices of simplex within a fixed range where 6 represents the dimension of vertex vector. In this study, the 7 initial vertices are denoted as $X_0, X_1, \ldots, X_6$.

Step1: Two procedures are performed in this step.
(1) Evaluation: Based on each vertex of simplex, we can compute the corresponding rigid transformation matrix defined in Eq. (1). According to the transformation matrix, the input point clouds yielded by coarse alignment are mapped into new coordinates. The new point clouds are forwarded into the trained TNFN to get distance function

$f(x, y, z)$. Then, $\sum f(x, y, z)$ can be calculated by sum of all mapping of new point clouds.

(2) Sorting: Here we choose total distance function $\sum f(x, y, z)$ as the cost function and re-define the symbol to be$C(X_i)$ where X_i indicates the i-th vertex of simplex. Then we sort the $C(X_i)$ and set the order as follows:

$$C(X_0) < C(X_1) < \ldots < C(X_6) \tag{8}$$

Step2: In this step, the reflection point X_6^R is calculated. The downhill simplex optimization utilizes the reflection point as the first candidate point to replace the worst point X_6. The reflection point is calculated as follows:

(a) First find centroid of the remaining point ($X_0 \sim X_5$):

$$M = \frac{1}{6}\sum_{i=1}^{5} X_i. \tag{9}$$

(b) Then seek the reflection point:

$$X_6^R = M + \alpha(M - X_6), \tag{10}$$

where $\alpha > 0$ and the default value is $\alpha = 1$.

(c) Finally, $C(X_6^R)$ can be calculated by the means of evaluation method described in Step 1.

Step3: There are 3 cases are discussed in this step.

Case 1. : If $C(X_6^R) \geq C(X_0)$ and $C(X_6^R) < C(X_5)$, choose X_6^R to replace X_6. Then we re-sort the simplex and forward to Step 4.

Case 2. : If $C(X_6^R) < C(X_0)$, compute the expansion point X_6^E as follows:

$$X_6^E = X_6^R + \gamma(X_6^R - M), \tag{11}$$

where $r > 0$ and the default value is $r = 1$. Then calculate the $C(X_6^E)$. If $C(X_6^E) < C(X_0)$, choose X_6^E to replace X_6. Otherwise, choose X_6^R to replace X_6. After that, we re-sort the simplex and forward to Step 4.

Case 3. : If $C(X_6^R) \geq C(X_0)$ and $C(X_6^R) \geq C(X_5)$, compute the contraction point X_6^C as follows:

$$X_6^C = M + \beta(\overline{X}_6 - M), \tag{12}$$

where $0 < \beta < 1$ and the default value is $\beta = 0.5$. If $C(X_6^R) < C(X_6)$, then $\overline{X}_6 = X_6^R$. Otherwise, if $C(X_6^R) \geq C(X_6)$, then $\overline{X}_6 = X_6$. Subsequent, check the case of $C(X_6^R)$ as follows:

(i) If $C(X_6^C) < C(X_6)$, choose X_6^C to replace X_6. Then, we re-sort the simplex and forward to Step 4.

(ii) If $C(X_6^C) \geq C(X_6)$, shrink the whole simplex toward X_0. After shrinking, the new simplex is expressed as:

$$[X_0, (\eta X_0 + (1-\eta)X_1), \ldots, (\eta X_0 + (1-\eta)X_5), (\eta X_0 + (1-\eta)\overline{X}_6)], \tag{13}$$

where $0 < \eta < 1$ and the default value is $\eta = 0.5$.

Step4: If the least cost function meet one of the following conditions, the downhill simplex method is terminated, and output the final results.

(a) The number of loops reaches a predefined maximal iteration value.

(b) The value of cost function is less than a minimal threshold.

Otherwise, if the least cost function does not meet the above conditions, then we feedback to the Step 2 to continue the optimization procedure.

To sum up, the final results of the downhill simplex method would output the best vertex of simplex. Then we decode it to the six degrees of freedom $(\phi, \varphi, \theta, x, y, z)$. These parameters can be transfer to a rotation matrix R and translation vector T. The fine alignment of 3D surface is completed by computing the rigid body transformation according the R and T.

3. Experimental results

In the current section, a vehicle model depicted in Fig. 2 is selected as a reference model. The reference model is constructed by 4907 point clouds which are uniformly distributed on its surface. Thus, the aim of the 3D surface alignment task defined in the experiment is to align the arbitrary input point clouds with the reference model.

The experimental results comprise two parts. The first part uses the synthesized point cloud sets to test the proposed alignment approach. In the second part, real 3D point cloud data scanned by a 3D imaging laser scanner are used to validate the alignment accuracy. In both parts of the experiments, the alignment algorithm is compared with the neural network method (NNM) [2] and ICP [13] to demonstrate superior performance of the proposed scheme.

A. Testing using synthesized 3D point cloud data

To perform the coarse alignment learning, 2000 synthesized point cloud sets are generated randomly within the range described in Table 1. For training the TNFN, 70% of point clouds (1400) are prepared for training data set and the remaining 30% of point clouds (600) are prepared for testing data set. The learning parameters for the TNFN training are defined in the left side of Table 2. The parameters of DMELA are cited from the original paper [19]. Thus, after the coarse alignment learning completes, the output of TNFN is an estimated pose that coarsely aligns the input points with the reference model.

In the TNFN-based surface modeling, we produce a cube model with the size of $5 \times 5 \times 5$ m that encloses the entire reference model. Within the cube model, 64,000 point clouds are uniformly sampled according the resolution setting (0.125 m). Thus, the sampled point clouds are utilized for training TNFN to model the reference surface. The learning parameters of the TNFN-based surface modeling are defined in the right side of Table 2. The parameters $(\alpha, \beta, \gamma, \eta) = (1, 0.5, 1, 0.5)$ are determined by practical experimentation. Once the training of TNFN-based surface modeling is completed, the TNFN modeling is combined with the downhill simplex optimization method to execute the fine alignment of 3D surface.

Because the execution time and alignment accuracy are two major issues for a surface alignment system, these elements are taken as the evaluation conditions to examine the proposed alignment system.

Table 1
Range of 3D rigid transformation parameters.

3D rigid transformation parameter	The range of affine transformation parameter
ϕ(degree), for roll	[−10 10]
φ(degree), for yaw	[−90 90]
θ(degree), for pitch	[0 90]
x(m)	[−0.2 0.2]
y(m)	[−0.2 0.2]
z(m)	[−0.2 0.2]

(1) Alignment accuracy
To evaluate the alignment accuracy, the proposed system is compared with NNM [2] and ICP [13] two methods that use PCA for coarse alignment. Therefore, based on the 600 testing sets of point clouds, the alignment errors of the coarse and fine alignments are listed in Table 3 where RMSE indicates the root mean square error. From this table, the proposed system exhibits the lowest coarse and fine alignment errors among all systems. In addition, the proposed method improves the PCA coarse alignment, as shown in the table. Fig. 8(a) and (b) presents a coarse alignment example of PCA and the proposed TNFN-based method, where the blue and red point clouds represent the testing and reference model data, respectively. From this figure, the proposed method results in less alignment error than the PCA method.

(2) Alignment speed
In consideration of alignment speed, the average execution time for aligning 600 testing sets of point clouds is calculated. The results of the alignment speed are also listed in Table 3. As shown in the table, the execution time of the proposed system is shorter than those of NNM and ICP. NNM and ICP are two methods that use PCA for coarse alignment. However, PCA cannot ensure that the laser scanned point clouds have the same orientation of principal axes as the reference model such that it would cause a high alignment error in the coarse alignment phase. To this end, the initial alignment error of NNM and ICP will be larger than our proposed alignment method. Such phenomenon results in large number of iteration to fine alignment procedure for NNM and ICP. Thus, the execute time of the proposed alignment method would be shorter than NNM and ICP.

B. Validation of real 3D point cloud data alignment

Fig. 9 represents a real case of 3D point cloud data scanned by a 3D imaging laser scanner. The size of the scanned scene is 256×256 with 20 degree field of view. In the 3D scenery, the

Table 2
Learning parameters for the TNFN training.

Parameters of training the TNFN	Value for coarse alignment	Value for surface modeling
Number of Fuzzy rules	20	40
$(m_{\min}, m_{\max})$	$[-15, 15]$	$[-2, 2]$
$(\sigma_{\min}, \sigma_{\max})$	[12,14]	[0.3, 0.9]
N-bin for VFH	72	Not used
Population size	50	80
Crossover rate	0.6	0.75
Mutation rate	0.2	0.05

Table 3
Results of alignment accuracy and execution time.

Method	Average RMSE (m)		Average execution time (s)
	Coarse alignment error (m)	Fine alignment error (m)	
TNFN-based coarse-to-fine alignment	**0.1022**	**0.0635**	**3.24**
PCA coarse alignment NNM fine alignment	0.2845	0.1427	4.38
PCA coarse alignment ICP fine alignment	0.2845	0.0688	49.48

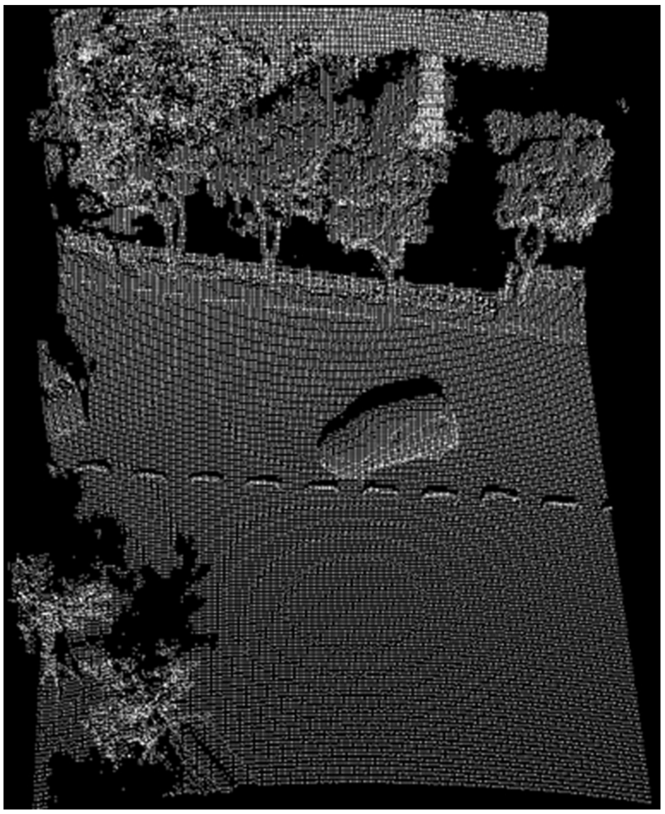


Fig. 9. Real case of 3D point cloud data scanned by a 3D imaging laser scanner.

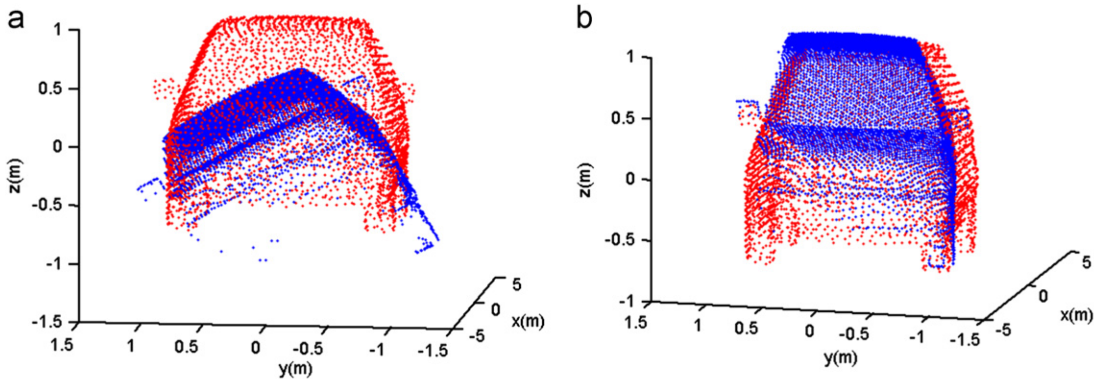


Fig. 8. Examples of two coarse alignment methods: (a) PCA and (b) TNFN-based coarse alignment.

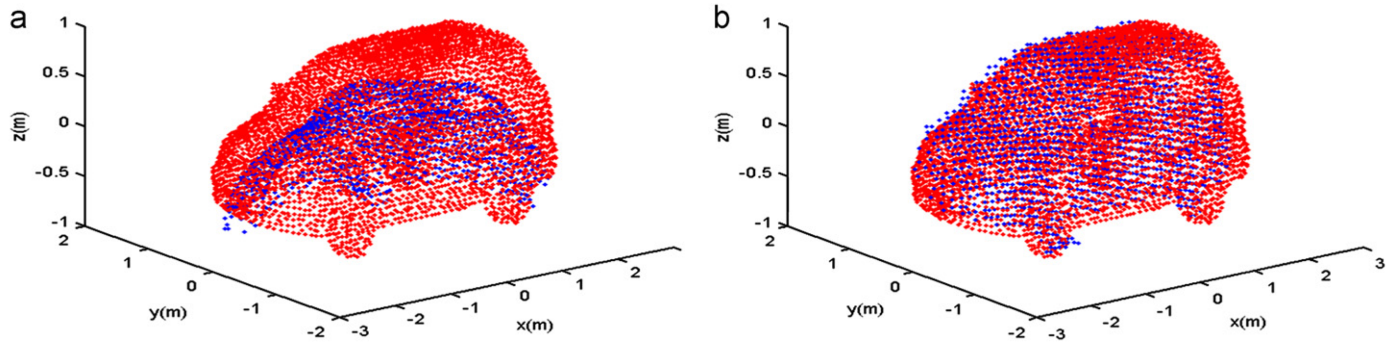


Fig. 10. Coarse alignment results: (a) PCA and (b) proposed TNFN-based method.

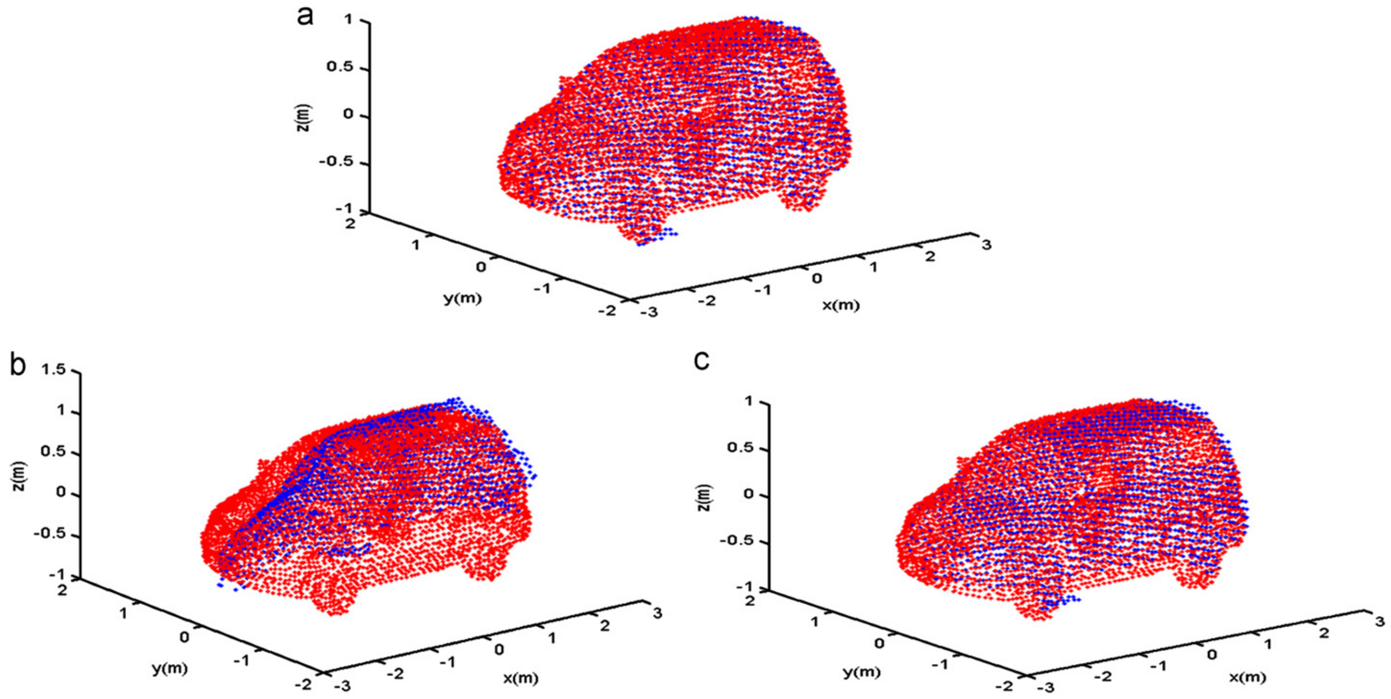


Fig. 11. Fine alignment results: (a) proposed TNFN-based method, (b) NNM, and (c) ICP.

vehicle region is extracted by using the segmentation algorithm described in [20]. The extracted vehicle data is then used to validate the alignment performance of the proposed system, NNM and ICP. Fig. 10(a) and (b) shows the coarse alignment results of PCA (used for NNM [2] and ICP [13] in coarse phase) and the proposed TNFN-based method. In this figure, the coarse alignment errors of PCA and the proposed approach are 0.262 and 0.106 m, respectively. Thus, this result again proves that the proposed method is superior to PCA. In the fine alignment case, Fig. 11(a)–(c) depicts the fine alignment results of proposed method, NNM, and ICP. From this figure, the fine alignment errors of the proposed system, NNM, and ICP are 0.0558, 0.1121, and 0.0569 m, respectively. These results indicate that the proposed TNFN-based method can achieve high accuracy in real 3D point cloud data. Furthermore, regarding the alignment speed, the execution time of the proposed system, NNM, and ICP are 1.71, 2.13, and 7.93 s, respectively. Therefore, the proposed system demonstrates higher alignment speed compared to NNM and ICP. In short, the proposed surface alignment system can align 3D point cloud data with the reference model accurately at high speed.

4. Conclusion

In this paper, a TNFN-based coarse-to-fine strategy has been proposed to address the problem of 3D surface alignment. In the coarse alignment phase, this study offers a TNFN-based pose estimation method to achieve higher alignment accuracy in comparison with the traditional PCA coarse alignment strategy. In the fine alignment phase, the proposed method combines TNFN-based surface modeling with the downhill simplex method to demonstrate lower alignment error compared to NNM and ICP. Furthermore, in consideration of alignment speed, the proposed coarse-to-fine strategy exhibits shorter execution time than other two methods. The evidence can be found in the experimental results of both synthesized and real laser scanned 3D surface. These findings are helpful for developing accurate and efficient 3D surface alignment systems.

Although the proposed model can obtain good alignment results, it still has a limitation. Specifically, in spite of combing the surface modeling with the downhill simplex optimization can obtain good results in fine alignment phase, the downhill simplex optimization may suffer from getting in local minima. Toward this end, the on-line parallel search techniques may be the solution for preventing the local minima happened. The on-lien parallel search techniques should be fast and keep the proper accuracy for applying to the fine alignment task. Therefore, the future work would investigate a parallel search to satisfy the design of the fine alignment phase.

Acknowledgement

This work was funded by Contract NSC 99–2623-E-009-006-D form the National Science Council, Taiwan, R.O.C.

References

[1] Liu H, Yan J, Zhang D. Three-dimensional surface registration: a neural network strategy. Neurocomputing 2006;70:597–602.

[2] Zhang J, Ge Y, Ong SH, Chui CK, Teoh SH, Yan CH. Rapid surface registration of 3D volumes using a neural network approach. Image Vision Comput 2008;26:201–210.

[3] Ghafoor, A, Iqbal, RN, and Khan, S, Robust image matching algorithm, 4th EURASIP Conference Focused on Video/Image Processing and Multimedia Communications, pp. 155–160, 2003.

[4] Chetverikov D, Stepanov D, Kresk P. Robust Euclidean alignment of 3D point sets: the trimmed iterative closet point algorithm. Image Vision Comput 2005;23:299–309.

[5] Liu YH. Improving ICP with easy implementation for free-form surface matching. Pattern Recognit 2004;37:211–226.

[6] Jack JJ, Roux C. Registration of 3D images by genetic optimization. Pattern Recognit Lett 1995;16:823–841.

[7] Malassiotis S, Strintzis MG. Snapshots: a novel local surface descriptor and matching algorithm for robust 3D surface alignment. IEEE Trans Pattern Anal Mach Intell 2007;29:1285–1290.

[8] Johan H, Li B, Wei Y, Iskandarsyah. 3D model alignment based on minimum projection area. Visual Comput 2011;27:565–574.

[9] Li B, Johan H. View context based 2D sketch-3D model alignment. In: Proceedings of IEEE workshop on applications of computer vision. p. 45–50, 2011.

[10] Werghi, N and Naqbi, MK, 3D face matching: an alignment approach using ordered facial surface patterns, in Proceedings of International Conference and Workshop on Current Trends in Information Technology, pp. 12–18, 2011.

[11] Werghi, N and Naqbi, MK, A novel surface pattern for 3D facial surface encoding and alignment, in Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 902–908, 2011.

[12] Alpert NM, Bradshaw JF, Kennedy D, Correia JA. The principal axes transformation – a method for image registration. J Nucl Med 1990;31:1717–1722.

[13] Besl P, Mckay N. A method for registration of 3D shapes. IEEE Trans Pattern Anal Mach Intell 1992;14(2):239–256.

[14] Peng, J, Strela, V, and Zorin, D, A simple algorithm for surface denoising, in Proceedings of IEEE Visualization, San Diego, CA, USA, pp. 107–112, 2001.

[15] Rusinkiewicz, S and Levoy, M, Efficient variants of ICP algorithm, in Prcoceedings of Third International Conference on 3D Digital Imaging and Modeling, Quebec City, Canada, pp. 145–152, 2001.

[16] Blais G, Levine M. Registering multiview range data to create 3D computer objects. IEEE Trans PAMI 1995;**17**(8):820–824.

[17] Wu CC, Lin SF. Efficient model detection in point cloud data based on bag of words classification. J Comput Inf Syst 2011;12:4170–4177.

[18] Rusu, RB, Bradski, G, Thibaux, R, and Hsu, J, Fast 3D recognition and pose using the viewpoint feature histogram, IEEE International Conference on Intelligent Robots and Systems, pp. 2155–2162, 2010.

[19] Chi-Yao Hsu Yi-Cheng, Lin Sheng-Fuu. Efficient and accurate image alignment using tsk-type neuro-fuzzy network with data-mining based evolutionary learning algorithm, EURASI. J Adv Sig Proc 2011;96.

[20] Rabbani, T., van den Heuvel, F.A., and Vosselmann, G., Segmentation of point clouds using smoothness constraint, in Proc. of ISPRS, 35, 248–253, 2006.